Typescript : Its a superset of JS

In typescripting strong typing is optional.

It consists of OOPs concepts

can catch error at compile time

we can get access to great tools

Intellisense

compiler compiles typescript file into JS file.

What is angular :

Its a frame work for building client application in html, css and either in javascript or in type script.

Javascript is ahrd to test.

Angular giving our appllication a claen and loosely coupled structure.

It is easy to understand and easy to maintain.

Includesvarious reusable code.

Application is more testable.

We can use automated test.

Architecture of angualr app:

npm install -g @angular/cli

to crate new project : > ng new hello-world

nodejs is a runtime environment for executing javascriptcode outside the browser

Node Package Manager :Used to install third pary libraries

Angular CLI :Cmd line interface

To create a new project :

In cmd -> go to the preferred location ->

> ng new hello-world

code editor: visualstudio.code

in VSC : go to the code location -> type code .

there we can see the folders

we use angular CLI to load our application in web server :

To do this

cmd > hello-world ng serve

here create a light cLI server listening to the localhost

Folders in Appln

e2e : End to end test tool

node_modules : contains third party libraries

src : Contains the source code of the application

assets : will store assets like image

Environment : Stores the configuration settings, One is for productionenvmnt and one for development envmt

main.cs ; the type script file , here starts the application

webpack : is a build automation tools

Hot Module REplacement : When the sorce code is modified webpack automatically refreshes the browser.

webpack automaticaly injects the scripts into index.html

Type script :

Its a supers et of JS

TS has additional features

TS typing is optional

OO features are there

can cathc error at compile time

acees to tools

intellisense

TS is a language

Build ->TS ->JS that browser can understand

To install typescript :

C:\AngularJS_Code> npm install -g typescript

Make directory :

C:\AngularJS_Code>mkdir ts-hello

code main.ts

in Main.ts

write some JS code:

```
function DoSomething() {
    for (var i = 0; i < 5; i++) {
        console.log(i);
    }
    console.log('Finally: ' + i);
}
DoSomething();
```

> code main.js will open the ts as js file

To compile with node : > node main.js

Versions of JS :

ECMA -all browsers support this version.

var , let is used to declarevariable

var : In the above code var is declare inside for but its accessible outside for

so var is accesssible to the nearest ouside function

but 'let' is accessible only to the nearest loop, like in c#

Rome file : >del main.js

Note : Eventhough there is compiletime errors in tsc, it will generate the JS file which can understnd the browser

ECMA code

Types in Type Script:

If we declare and define a let then the first type assigned will be the type

if we didnt define it can accept any type, thats not safe

so we use type annotation:

let a: number

let e: number[]

let f: any[]=[1,true,'aa']

Enum :

const Red=0;

const Green=1;

enum Color{Red,Green};

let BackgroundColor=Color.Green;

Type Assertions :

Here we can define the variable type, even thogh we didnt do that using type annotation or defining at declaration

```
let message;
message = 'abc';
let endsWithC = (<string>message).endsWith('c');
let alternativeWay = (message as string).endsWith('c');
```

Arrow Functions :

```
let log = function(message) {
    console.log(message);
}

let doLog = (message) => console.log(message);
```

Instead of using above function declaration we can use below function declaration.

Angular Fundamentals :

Building blocks of angular apps :

Component : It encapsulates the data , html markup and logic for a view

Angular uses component based architecture that helps us to work on smaller and more miantainable code that also can reuse in different places.

Every application has atleast one component which is app component

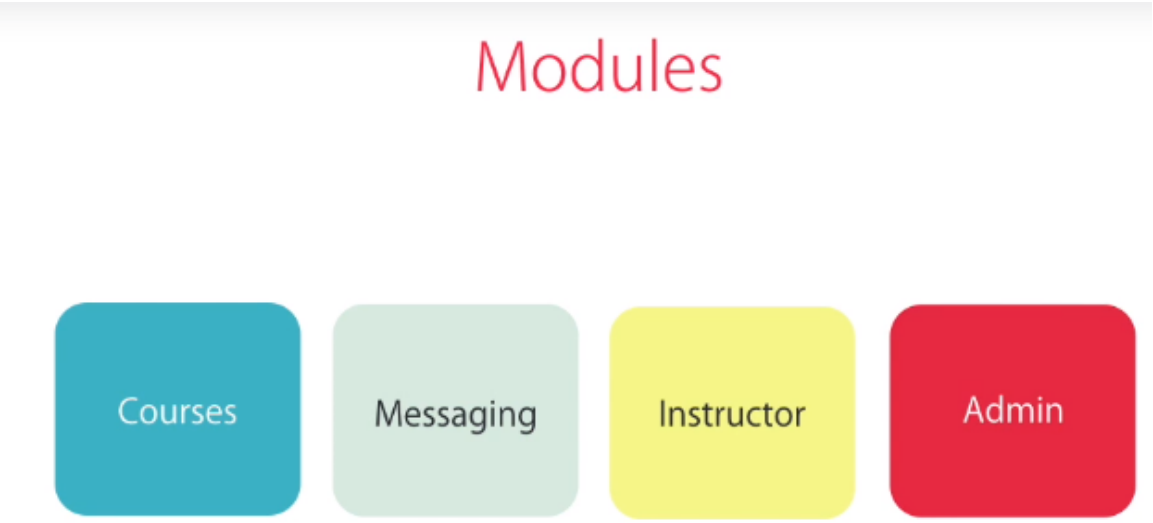
Every angular application is a tree of components.

Modules:

Module is a container for a group of related component.

Every angular app have atleast one module called app-module

Module example in udemy



Creating Components:

Steps to use component :

Create component

Register in module

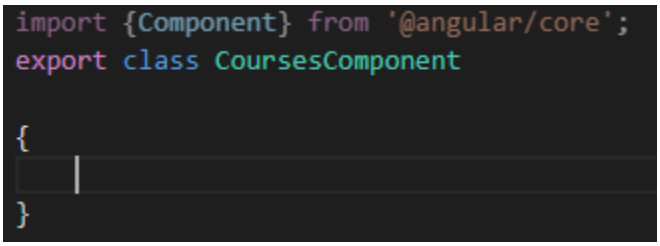
add an element in html markup.

In Explorer : src -> app -> add file -> courses.component.ts

Craete a class

Inorder to angular see this class first we need to export it

Inorder to make this code a component we need to add metadate, so we need to add decorator



To add component decorator we need to import Component namespace

import {Component} from '@angular/core'

Component is the namespace

angular/core is the library.

creating the decorator

//its a decoratoer function

@Component ({

})

class ->

id -> #

decorator function properties

selector: 'courses' //used to select element

// when we refer the selector it will add the corresponding component

template : '<h2>Courses </h2>'

// it is the markup rendering for this component

//In real world application template will be a large code , so we usually creatye another file fpr this.

courses.component.ts

```ts

import { Component } from '@angular/core';

@Component({
    selector: 'courses',
    template: '<h2>Courses</h2>'
})
export class CoursesComponent {
}
```

REgister this component in module

```ts
import { CoursesComponent } from './courses.component';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent,
    CoursesComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

Auto-impo

Here in @NgModule decorator in declarations propery we add our component CousesComponent

Then in the app.component.html

add :

```
<course></course>
```

Creating component using ng:

This is tedious step, so we can use ng to create component

>ng g c course

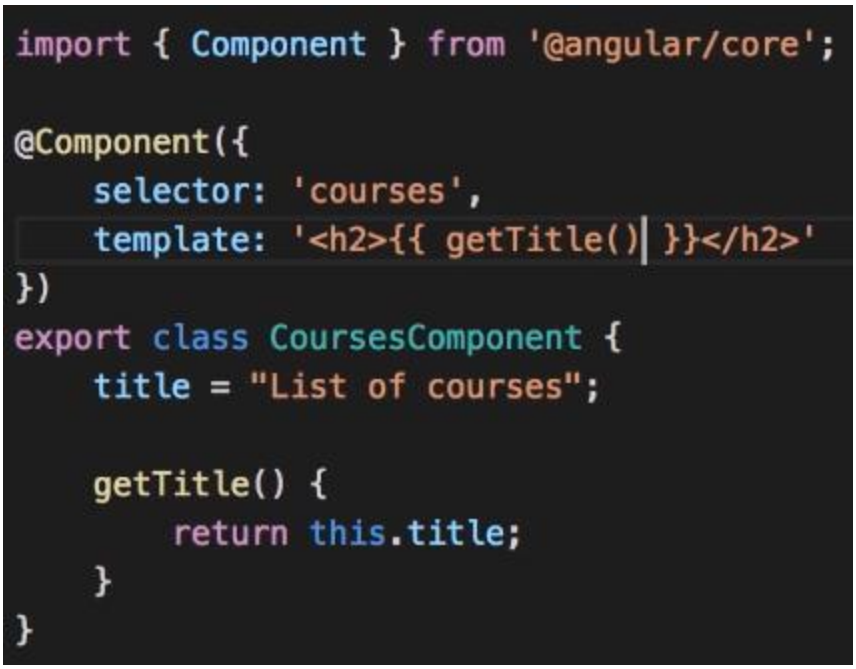
```
import { Component } from '@angular/core';

@Component({
    selector: 'courses',
    template: '<h2>{{ title }}</h2>'
})
export class CoursesComponent {
    title = "List of courses";
}
```

here value of the title isevaluated at runtime and adding to the DOM

If value of title changes at future it will automatically updated in the DOM

This is called databinding.



```
import { Component } from '@angular/core';

@Component({
    selector: 'courses',
    template: '<h2>{{ getTitle() }}</h2>'
})
export class CoursesComponent {
    title = "List of courses";

    getTitle() {
        return this.title;
    }
}
```

here we creating a method and calling that in the template.

class CourseComponent

{

title="List of courses";

getTitle()

{

return this.title;

}

}

string interpolation

Directive :

Is used to manipulate Dom element.

```
import { Component } from '@angular/core';

@Component({
    selector: 'courses',
    template: `
        <h2>{{ title }}</h2>
        <ul>
            <li *ngFor="let course of courses">
                {{ course }}
            </li>
        </ul>
    `
})
export class CoursesComponent {
    title = "List of courses";
    courses = ["course1", "course2", "course3"];
}
```